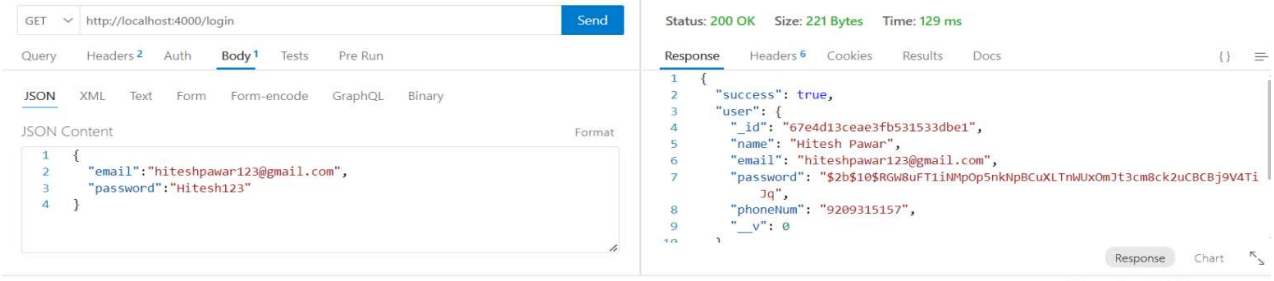
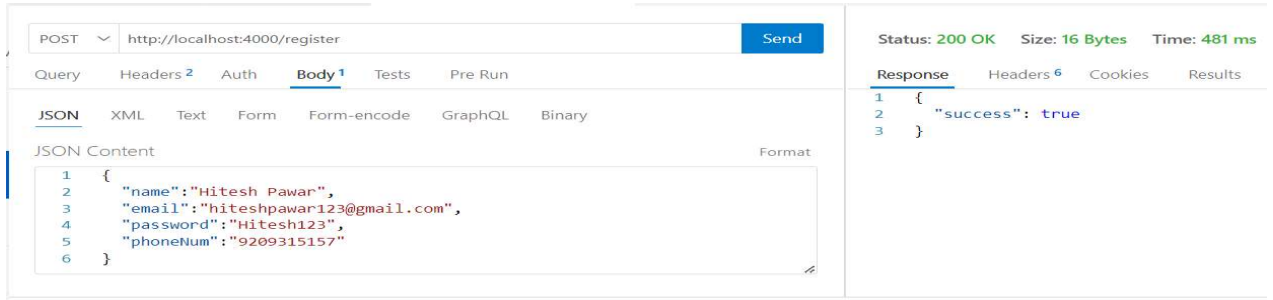
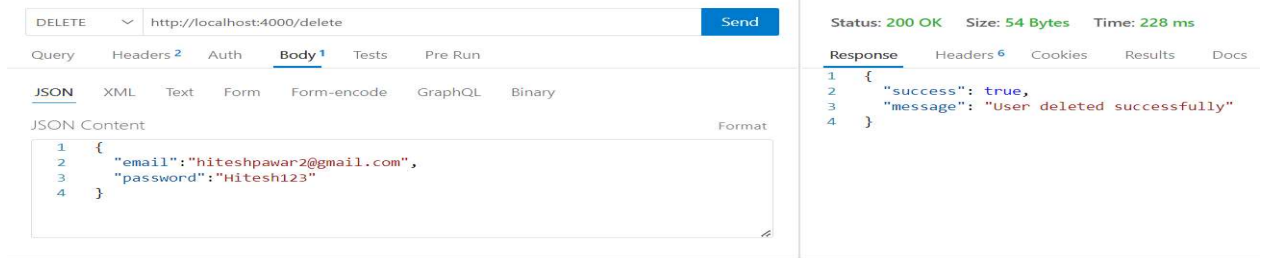
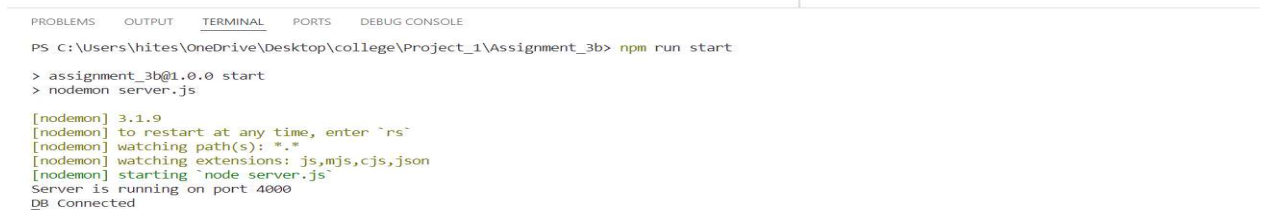

GET
http://localhost:4000/login
Send
Query
Headers 2
Auth
Body 1
Tests
Pre Run
JSON
XML
Text
Form
Form-encode
GraphQL
Binary
JSON Content
Format
{
"email":"hiteshpawar123@gmail.com",
"password":"Hitesh123"
}
Status: 200 OK
Size: 221 Bytes
Time: 129 ms
Response
Headers 6
Cookies
Results
Docs
{
"success": true,
"user": {
"_id": "67e4d13ceae3fb531533dbe1",
"name": "Hitesh Pawar",
"email": "hiteshpawar123@gmail.com",
"password": "$2b$10$RGW8uFT1iNMpOp5nkNpBCuXLTnWUxOmJt3cm8ck2uCBCBj9V4TiJq",
"phoneNum": "9209315157",
"__v": 0
Response
Chart

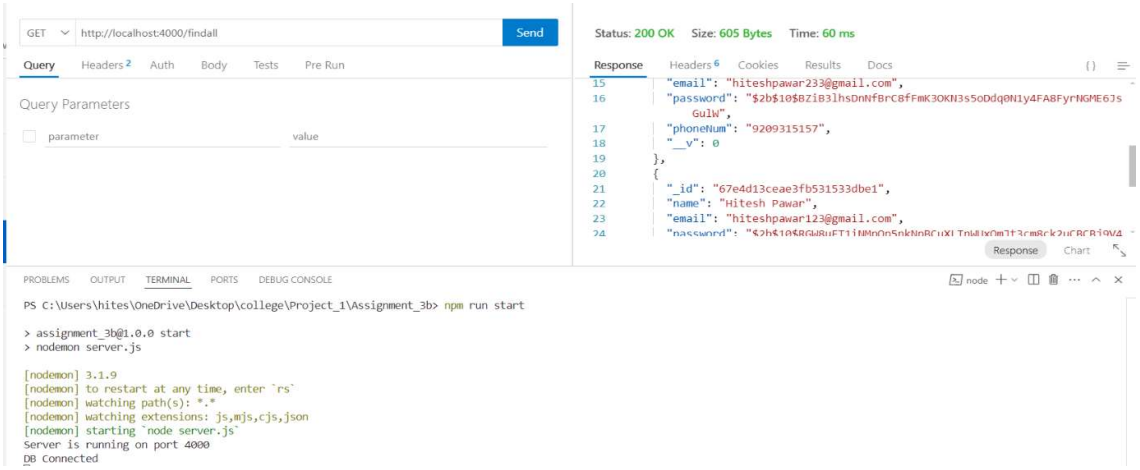

PROBLEMS
OUTPUT
TERMINAL
PORTS
DEBUG CONSOLE
node
PS C:\Users\hites\OneDrive\Desktop\college\Project_1\Assignment_3b> npm run start
> assignment_3b@1.0.0 start
> nodemon server.js
[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server is running on port 4000
DB Connected


POST
http://localhost:4000/register
Send
Query
Headers 2
Auth
Body 1
Tests
Pre Run
JSON
XML
Text
Form
Form-encode
GraphQL
Binary
JSON Content
Format
{
"name":"Hitesh Pawar",
"email":"hiteshpawar123@gmail.com",
"password":"Hitesh123",
"phoneNum":"9209315157"
}
Status: 200 OK
Size: 16 Bytes
Time: 481 ms
Response
Headers 6
Cookies
Results
{
"success": true
}


PROBLEMS
OUTPUT
TERMINAL
PORTS
DEBUG CONSOLE
PS C:\Users\hites\OneDrive\Desktop\college\Project_1\Assignment_3b> npm run start
> assignment_3b@1.0.0 start
> nodemon server.js
[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server is running on port 4000
DB Connected


DELETE
http://localhost:4000/delete
Send
Query
Headers 2
Auth
Body 1
Tests
Pre Run
JSON
XML
Text
Form
Form-encode
GraphQL
Binary
JSON Content
Format
{
"email":"hiteshpawar2@gmail.com",
"password":"Hitesh123"
}
Status: 200 OK
Size: 54 Bytes
Time: 228 ms
Response
Headers 6
Cookies
Results
Docs
{
"success": true,
"message": "User deleted successfully"
}


PROBLEMS
OUTPUT
TERMINAL
PORTS
DEBUG CONSOLE
PS C:\Users\hites\OneDrive\Desktop\college\Project_1\Assignment_3b> npm run start
> assignment_3b@1.0.0 start
> nodemon server.js
[nodemon] 3.1.9
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node server.js`
Server is running on port 4000
DB Connected



- **server.js**

```
import express from "express"
import { connectDB } from "./config/db.js";
import userRouter from "./routes/userRoutes.js";
const app=express();
app.use(express.json());
connectDB();
app.use("/",userRouter)
app.get("/",(req,res)=>{
   console.log("Api is Working");
});
app.listen(port,(req,res)=>{
   console.log(`Server is running on port ${port}`)
});
```

- **userModel.js**

```
import mongoose from "mongoose"
const userSchema = new mongoose.Schema({
      name:{type:String,required:true},
      email: { type: String, required: true, unique: true },
      password: { type: String,required:true},
      phoneNum: {type:String}
   });
const userModel =mongoose.models.user|| new mongoose.model('User', userSchema);
export default userModel;
```

- **userRoutes.js**

```
import express from "express"
import { deleteUser, findAll, loginUser, registerUser, updateUser } from "../controllers/userController.js";

const userRouter=express.Router();

userRouter.post("/register",registerUser);
userRouter.get("/findall",findAll);
userRouter.get("/login",loginUser);
userRouter.put("/update",updateUser);
userRouter.delete("/delete",deleteUser);

export default userRouter;
```

- **userController.js**

```
import validator from "validator";
import bcrypt from "bcrypt";
import userModel from "../models/userModels.js";

const registerUser = async (req, res) => {
  const { name, email, password, phoneNum } = req.body;

  try {
    const existingUser = await userModel.findOne({ email });
    if (existingUser) {
      return res.json({ success: false, message: "Duplicate User" });
    }

    if (!validator.isEmail(email)) {
      return res.json({ success: false, message: "Please enter a valid email" });
    }

    if (!validator.isAlpha(name, 'en-US', { ignore: ' ' })) {
      return res.json({ success: false, message: "Please enter a valid name" });
    }

    if (!validator.isMobilePhone(phoneNum, 'en-IN')) {
      return res.json({ success: false, message: "Please enter a valid 10-digit phone number" });
    }

    if (password.length < 8) {
      return res.json({ success: false, message: "Please enter a strong password (at least 8 characters)" });
    }

    const salt = await bcrypt.genSalt(10);
    const hashedPassword = await bcrypt.hash(password, salt);

    const newUser = new userModel({
      name,
      email,
      password: hashedPassword,
      phoneNum,
    });

    await newUser.save();
    res.json({ success: true });
  } catch (error) {
    console.error(error);
    res.json({ success: false, message: "Error registering user" });
  }
};

const loginUser = async (req, res) => {
  const { email, password } = req.body;

  try {
    const user = await userModel.findOne({ email });
    if (!user) {
      return res.json({ success: false, message: "User Not Found" });
    }

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.json({ success: false, message: "Invalid Credentials" });
    }

    res.json({ success: true, user });
  } catch (error) {
    console.error(error);
    res.json({ success: false, message: "Error logging in" });
  }
```

```
};

const updateUser = async (req, res) => {
  const { email, name, phoneNum ,password} = req.body;

  try {
    const user = await userModel.findOne({ email });
    if (!user) {
      return res.json({ success: false, message: "User Not Found" });
    }

    if (!validator.isAlpha(name, 'en-US', { ignore: ' ' })) {
      return res.json({ success: false, message: "Please enter a valid name" });
    }

    if (!validator.isMobilePhone(phoneNum, 'en-IN')) {
      return res.json({ success: false, message: "Please enter a valid 10-digit phone number" });
    }
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.json({ success: false, message: "Invalid Credentials" });
    }
    const updatedUser = await userModel.findOneAndUpdate(
      { email },
      { $set: { name, phoneNum } },
      { new: true, runValidators: true }
    );

    res.json({ success: true, user: updatedUser });
  } catch (error) {
    console.error(error);
    res.json({ success: false, message: "Error updating user" });
  }
};

const deleteUser = async (req, res) => {
  const { email ,password} = req.body;

  try {
    const user = await userModel.findOne({ email });
    if (!user) {
      return res.json({ success: false, message: "User Not Found" });
    }
    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) {
      return res.json({ success: false, message: "Invalid Credentials" });
    }
    const deletedUser = await userModel.findOneAndDelete({ email });
    res.json({ success: true, message: "User deleted successfully" });
  } catch (error) {
    console.error(error);
    res.json({ success: false, message: "Error deleting user" });
  }
};

const findAll = async (req, res) => {
  try {
    const users = await userModel.find();
    res.json({ success: true, users });
  } catch (error) {
    console.error(error);
    res.json({ success: false, message: "Error fetching users" });
  }
};

export { registerUser, loginUser, deleteUser, updateUser, findAll };
```